IN THE TITLE:

Please amend the title to read as follows: "Copy on Write File System Consistency and Block Usage." B1

IN THE SPECIFICATION:

Please amend the paragraph at page 5, lines 11 to 16, to read as follows:

As illustrated in Figure 1, every direct pointer 110A, 112A-112B, 120A, and 122A and indirect pointer 110B and 120B in the Episode file system contains a COW bit. Blocks that have not been modified since the clone was created are contained in both the active file system and the clone, and have set (1) COW bits. The COW bit is cleared (0) when a block that is referenced to by the pointer has been modified and, therefore, is part of the active file system but not the clone. B2

Please amend the paragraph at page 7, lines 9 to 22, to read as follows:

Creating a clone in the prior art can use up as much as 32 MB on a 1 GB disk. The prior art uses 256 MB of disk space on a 1 GB disk (for 4 KB blocks) to keep eight clones of the file system. Thus, the prior art cannot use large numbers of clones to prevent loss of data. B3

Instead it used to facilitate backup of the file system onto an auxiliary storage means other than the disk drive, such as a tape backup device. Clones are used to backup a file system in a consistent state at the instant the clone is made. By doping the file system, the clone can be backed up to the auxiliary storage means without shutting down the active file system, and thereby preventing users from using the file system. Thus, clones allow users to continue accessing an active file system while the file system, in a consistent state, is backed up. Then the clone is deleted once the backup is completed. Episode is not capable of supporting multiple clones since each pointer has only one COW bit. A single COW bit is not able to distinguish more than one clone. For more than one clone, there is no second COW bit that can be set.

B3

Please amend the paragraph at page 21, lines 4 to 11, to read as follows: 17 18 31

For a file size greater than 64 MB, the 16 buffer pointers of the inode reference double-indirect WAFL buffers. Each 4 KB double-indirect WAFL buffer comprises 1024 pointers pointing to corresponding single-indirect WAFL buffers. In turn, each single-indirect WAFL buffer comprises 1024 pointers that point to 4 KB direct WAFL buffers. Thus, up to 64 GB can be addressed. Figure 9D is a diagram illustrating a Level 3 inode 820 comprising 16 pointers 820B wherein pointers PTR0, PTR1, and PTR15 reference double-indirect WAFL buffers 970A, 970B, and 970C, respectively. Double-indirect WAFL buffer 970A comprises 1024 pointers that point to 1024 single-indirect WAFL buffers 980A-980B. Each single-indirect WAFL buffer 980A-980B, in turn, references 1024 direct WAFL buffers. As shown in Figure

B4

B4

9D, single-indirect WAFL buffer 980A references 1024 direct WAFL buffers 990A-990C and single-indirect WAFL buffer 980B references 1024 direct WAFL buffers 990D-990F.

Please amend the paragraph at page 23, line 9, to page 24, line 3, to read as follows:

A first met-data file is the "inode file" that contains inodes describing all other files in the file system. Figure 12 is a diagram illustrating an inode file 1210. The inode file 1210 may be written anywhere on a disk unlike prior art systems that write "inode tables" to a fixed location on disk. The inode file 1210 contains an inode 1210A-1210F for each file in the file system except for the inode file 1210 itself. The inode file 1210 is pointed to by an inode referred to as the "root inode". The root inode is kept in a fixed location on disk referred to as the file system information (fsinfo) block described below. The inode file 1210 itself is stored in 4 KB blocks on disk (or 4 KB buffers in memory). Figure 12 illustrates that inodes 1210A-1210C are stored in a 4 KB buffer 1220. For on-disk inode sizes of 128 bytes, a 4 KB buffer (or block) comprises 32 inodes. The incore inode file 1210 is composed of WAFL buffers 1220. When an incore inode (i.e., 820) is loaded, the on-disk inode part of the incore inode 820 is copied from the buffer 1220 of the inode file 1210. The buffer data itself is loaded from disk. Writing data to disk is done in the reverse order. The incore inode 820, which contains a copy of the ondisk inode, is copied to the corresponding buffer 1220 of the inode file 1210. Then, the

B5

51

B

B5

inode file 1210 is write-allocated, and the data stored in the buffer 1220 of the inode file 1210 is written to disk.

Please amend the paragraph at page ~~51~~ 39, lines ~~4~~ 8 to ~~12~~ 17, to read as follows:

Referring to Figure 22, two previous snapshots 2110A and 2110B exist on disk. At the instant when a third snapshot is created, the root inode pointing to the active file system is copied into the inode entry 2110C for the third snapshot in the inode file 2110. At the same time in the consistency point that goes through, a flag indicates that snapshot 3 has been created. The entire file system is processed by checking if BIT0 for each entry in the blkmap fife is set (1) or cleared (0). All the BIT0 values for each blkmap entry are copied into the plane for snapshot three. When completed, every active block 2110-2116 and 1207 in the file system is in the snapshot at the instant it is taken.

B6
